Renesas RA Family

Getting Started with Low Power Applications

Introduction

This Application Note describes how you can reduce the effective power consumption of the RA Microcontroller using Low Power Modes (LPMs). Two accompanying application projects show common use cases of entering Low Power Modes and configuring the various peripherals to exit the entered mode. Upon completion of this guide, you will be able to add an LPM module to your own design, configure it correctly for the target application, and write code using the included application project as a reference and efficient starting point.

- This application note describes LPM module usage in different modes and supported peripherals
- Application overview for the different use cases
- FSP configuration steps for LPM
- Application design highlights
- Importing, loading, and running the application project
- Project migration steps to other RA Kits.

Required Resources

- e² studio ISDE v20.7.0 or later
- Flexible Software Package (FSP) v1.3.0 or later
- J-Link RTT viewer V6.60 or later

Primary target devices

- EK-RA6M3 kit

Table 1. RA Kits Tested with LPM Application

<table>
<thead>
<tr>
<th>Kit</th>
<th>Operable Long Timer in LPM</th>
<th>LPM Transition and Clock Changing at Run-Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>EK-RA6M3</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>EK-RA6M2</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>EK-RA6M1</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>EK-RA4M1</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>EK-RA2A1</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
1. Application Overview ................................................................................................................. 4
   1.1 Low Power Modes .................................................................................................................. 4
      1.1.1 Sleep Mode ...................................................................................................................... 5
      1.1.2 Software Standby Mode .................................................................................................. 5
      1.1.3 Snooze Mode ................................................................................................................... 5
      1.1.4 Deep Software Standby Mode .......................................................................................... 5
      1.1.5 Activation and Cancel Sources .......................................................................................... 6
   1.2 Peripheral Operation in LPM ................................................................................................... 6
   1.3 Use Case: Changing Clocks at Run-Time .................................................................................. 6
   1.4 Use Case: LPM Transition at Run-Time ................................................................................... 6
   1.5 Use Case: Operable Long Timer in Software Standby and Deep Software Standby Modes ...... 7

2. LPM HAL Module .......................................................................................................................... 7

3. FSP Configuration ....................................................................................................................... 8
   3.1 Components Tab .................................................................................................................... 9
   3.2 Stacks Tab ............................................................................................................................. 10
   3.3 Module Configuration ........................................................................................................... 11
      3.3.1 LPM Configuration .......................................................................................................... 11
         3.3.1.1 Activation and Cancelation Sources ............................................................................ 11
         3.3.1.2 Sleep Mode Configuration .......................................................................................... 11
         3.3.1.3 Software Standby Mode Configuration .......................................................................... 12
         3.3.1.4 Snooze Mode Configuration ....................................................................................... 12
         3.3.1.5 Deep Software Standby Configuration ........................................................................... 13
      3.3.2 Timer Configuration ......................................................................................................... 14
         3.3.2.1 RTC Configuration .................................................................................................... 14
         3.3.2.2 AGT Timer Configuration ............................................................................................ 14
   3.4 Pin Configuration ................................................................................................................... 15
      3.4.1 Pin Configuration in Normal Mode .................................................................................. 16
      3.4.2 Pin Configuration in LPM .............................................................................................. 16

4. Application Architectures ............................................................................................................. 17
   4.1 Clock Changing and LPM Transition ....................................................................................... 18
   4.2 RTC Timer Operation in LPM ............................................................................................... 20
   4.3 Operable Long Timer in Software Standby and Deep Software Standby Modes ................... 20

5. Application Code Highlights ...................................................................................................... 21
   5.1 Clock Source Setup ............................................................................................................... 21
   5.2 Handle On-Chip Modules in LPM to Reduce Power Consumption ......................................... 21
   5.3 Change System Clock at Run-Time ......................................................................................... 23

6. Importing and Building the Project .......................................................................................... 24
7. Running Applications ............................................................................................................. 24
   7.1 Board Setups ....................................................................................................................... 24
   7.2 Downloading the Executables ............................................................................................... 24
   7.2.1 Using a debugging interface with e² studio ........................................................................ 24
   7.2.2 Using J-Link tools ............................................................................................................ 25
   7.2.3 Using Renesas Flash Programmer ....................................................................................... 25
   7.3 User Interface ..................................................................................................................... 25
       7.3.1 LED Indication ................................................................................................................. 25
       7.3.1.1 Clock Changing and LPM Transition ............................................................................ 25
       7.3.1.2 Operable Long Timer ................................................................................................... 25
       7.3.2 User Push Button Input .................................................................................................. 25
           7.3.2.1 Clock Changing and LPM Transition ........................................................................ 25
           7.3.2.2 Operable Long Timer ................................................................................................. 25
       7.3.3 RTT Console .................................................................................................................. 25
   7.4 Debugging Low Power Modes ............................................................................................... 26
   7.5 Steps to Run the Application ................................................................................................ 27
       7.5.1 Clock Changing: ............................................................................................................. 27
       7.5.2 LPM Transition ............................................................................................................... 27
       7.5.3 Operable Long Timer .................................................................................................... 28
   7.6 Measure MCU Current ......................................................................................................... 29
   8. Migrating LPM Applications to Different MCU/Kit ............................................................. 29
   9. References ............................................................................................................................ 35

Revision History .......................................................................................................................... 37
1. Application Overview

Application Projects accompanying this document serve as references to operate the microcontroller (MCU) in various Low Power Modes demonstrating different levels of power consumption often required to maximize battery life.

For ease of understanding the LPM, these application projects covers the different Low Power Modes with different clock settings to showcase the each mode, operation of different peripherals in an LPM, required pin configurations, trigger/end source configuration, and a user interface to initiate transition to different LPM states and switch back to Normal mode. The configuration for each mode is maintained as an independent instance. Users can use these example configurations and change different settings to trigger/end operation as desired.

In addition to the LPM, the application also supports changing the source clock of the MCU dynamically and running LPM for these clocks.

1.1 Low Power Modes

RA MCU’s support four different types of LPM depending on the MCU family. These are:

- Sleep mode
- Software Standby mode
- Snooze mode
- Deep Software Standby mode.

Low power mode transition and triggering sources for RA MCUs are illustrated in Figure 1. For more details on these transitions, see the User’s Manual for the specific MCU.

![Figure 1. LPM Transition Diagram for RA6M3](image-url)
In LPM, the CPU stops, but on-chip peripherals and oscillator states may be operational depending on the LPM selected. Therefore, their effects on MCU power consumption are very different. The typical current consumption when the MCU is in a Low Power Mode is found in the MCU Hardware User’s Manual section on Operating and Standby Current. Figure 2 shows the typical power consumption when the MCU is in a Low Power Mode vs throughput.

![Figure 2. Power Consumption and Throughput of the LPM](image)

In order for the MCU to enter or exit the LPM, associated special function registers need to be configured. This app note doesn’t focus on the bit-level configuration details, since the bits can be configured using the API provided by the FSP. The API provided by the FSP is documented in the FSP User’s Manual. If you want to explore more details on the LPM and its supported list of peripherals, interrupts, refer to the Low Power Modes section in the RA MCU Datasheet. Low Power Modes commonly available with RA MCUs are described next.

1.1.1 Sleep Mode
An operational CPU is typically the primary cause of power consumption. In Sleep mode, the CPU stops operating, but the contents of its internal registers are retained. Other peripheral functions in the MCU do not stop. Available resets or interrupts in Sleep mode can cause the MCU to cancel Sleep mode. All interrupt sources are available in this mode to cancel the Sleep mode. When using an interrupt to successfully cancel Sleep mode, you must set the associated IELSRn register before executing a WFI instruction.

1.1.2 Software Standby Mode
In Software Standby mode, the CPU, most of the on-chip peripheral functions and the oscillators stop operation. However, the contents of the CPU internal registers and the SRAM data, the states of the on-chip peripheral functions, and the I/O port states are retained. Software Standby mode allows a significant reduction in power consumption since most of the oscillators stop in this mode.

1.1.3 Snooze Mode
The Snooze feature provides operational flexibility to dramatically reduce current consumption. Snooze is an extension to the Software Standby mode where limited peripheral modules can operate without waking up the CPU. The Snooze mode can be entered through the Software Standby mode via configured interrupt sources and similarly woken up from Snooze mode by interrupts supported in the Snooze mode.

1.1.4 Deep Software Standby Mode
In Deep Software Standby mode, the CPU and on-chip peripheral functions except for some internal modules are stopped. Current consumption is reduced since the internal power supply to these modules is stopped prior to entering Deep Software Standby mode. The contents of all of the CPU registers and internal peripheral modules, except for the RTC alarm, RTC interval, and USB suspend/resume detecting unit, become undefined.
1.1.5 Activation and Cancel Sources

Low power modes are canceled by various interrupt sources such as RES pin reset, power-on reset, voltage monitor reset, and peripheral interrupts. Refer to the Low Power Modes section in Renesas RA MCU User’s Manual for list of interrupt sources for different LPMs.

For Deep Software Standby mode, the reset cause can be identified via the DPSRSTF flag bit in the Reset Status Register0 (RSTSR0).

Only Snooze mode is triggered by a Snooze request to enter snooze mode from Software Standby mode. The transitions to other LPMs are done by executing a WFI instruction with appropriate settings in the Standby Control register (SBYCR).

1.2 Peripheral Operation in LPM

Not all the MCU peripherals are available in different LPMs. MCU peripherals also have different setting retention capabilities during the different LPMs, such as contents of the internal registers may be retained in some LPMs, but the contents may be undefined in other modes. Depending upon the application, users are required to choose the peripherals and LPM for the maximum power savings. Users are also required to turn off/disable oscillators and on-chip peripherals that are not clock gated or powered off to maximize the power savings. Refer to the Low Power Modes section in each RA MCU User’s Manual: Hardware to understand different oscillator and peripherals available in a specific LPM.

In the next sections we will talk about the use case scenarios for the different LPMs with different clock settings and peripherals.

1.3 Use Case: Changing Clocks at Run-Time

This application use case describes how to change the RA MCU clock dynamically and set it to different clock settings supported by the RA MCU using the FSP CGC HAL driver APIs. While the user can configure the Clock Generation Circuits (CGC) within the MCU using the RA FSP Clock Configurator, in many applications, where the MCU is eventually powered by a battery, there is an inherent requirement to change the clock configuration settings as the MCU is running. Based on the desired set of clock sources, MCU changes to different clock source and operates normally without reboot.

Changing the system clock affects the peripherals, which use derivatives of the system clock as a source and other clocks in the system. Users are advised to select the dividers as applicable for the system. When changing the clock, make sure that stabilization with the proper settling time is in place. This stabilization time is designed into the CGC HAL Driver.

This application operates using the user switch input to change the MCU clock mode from the previously running clock to the desired clock. The new clock settings are applied and displayed via the RTT interface for the user notification.

Table 2 shows the available user selectable clock settings in the application

### Table 2. User Selectable Clock

<table>
<thead>
<tr>
<th>Clock Source</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOSC</td>
<td>Main Oscillator Clock</td>
</tr>
<tr>
<td>HOCO</td>
<td>High speed on-chip Oscillator</td>
</tr>
<tr>
<td>MOCO</td>
<td>Medium Speed on-chip Oscillator</td>
</tr>
<tr>
<td>LOCO</td>
<td>Low Speed on-chip Oscillator</td>
</tr>
<tr>
<td>SOSC</td>
<td>Sub Oscillator Clock</td>
</tr>
</tbody>
</table>

The sequence of the clock being configured are from MOSC→HOCO→MOCO→LOCO→SOSC→MOSC. The objective of this use case to show the different clock sources which can be changed during run time without halting the MCU. Changing the clock dynamically is accomplished by using the RA CGC HAL driver API R_CGC_ClocksCfg. For more details on CGC HAL driver API, refer to the FSP User’s Manual.

1.4 Use Case: LPM Transition at Run-Time

This use case shows the different LPMs supported by the MCU for the different clock settings.
The application requires user push button switch input to change the LPM available for the MCU and perform transitions as programmed. The supported LPM and its transitions to the different LPMs are displayed using the RTT interface for notifying the user. The application also showcases the use of few peripherals, like the AGT timer, and RTC operating in different LPMs and displaying the RTC time information regularly when MCU transitions to the normal mode from the LPM. AGT1 Timer is used in the Snooze mode to alternate between Software standby mode and Snooze mode. RTC Alarm interrupt is used to cancel the Software Standby mode and enter normal mode. IRQn (User Switch Interrupt) is used to cancel the Sleep mode and Deep Software Standby mode.

The visual indication of the LPM transition can also be seen with the User LED on the Board. When the LED is blinking approximately every 1 seconds, it is running in the Normal mode. If the LED is turned OFF, it is in an LPM.

Note: More details on the application are explained in the architecture section 4.1. The peripherals used in the application are just few of those available for the MCU. For the complete list of peripherals supported in the LPM, refer to the LPM section of the MCU datasheet.

Different clock sources and LPMs supported for the RA MCUs are shown in Table 3.

Table 3. Clock Sources and Supported LPM for RA MCUs

<table>
<thead>
<tr>
<th>Clock Source Supported</th>
<th>LPM Supported</th>
<th>LPM Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCU</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RA6M3, RA6M2</td>
<td>SLEEP, SW_STNDBY, SNOOZE, DEEP_SW_STANDBY</td>
<td>SLEEP, SW_STNDBY, SNOOZE, DEEP_SW_STANDBY</td>
</tr>
<tr>
<td>MOSC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HOCO</td>
<td>SLEEP, SW_STNDBY, SNOOZE, DEEP_SW_STANDBY</td>
<td>SLEEP, SW_STNDBY, SNOOZE, DEEP_SW_STANDBY</td>
</tr>
<tr>
<td>MOCO</td>
<td>SLEEP, SW_STNDBY, SNOOZE, DEEP_SW_STANDBY</td>
<td>SLEEP, SW_STNDBY, SNOOZE, DEEP_SW_STANDBY</td>
</tr>
<tr>
<td>LOCO</td>
<td>SLEEP, SW_STNDBY, SNOOZE, DEEP_SW_STANDBY</td>
<td>SLEEP, SW_STNDBY, SNOOZE, DEEP_SW_STANDBY</td>
</tr>
<tr>
<td>SOSC</td>
<td>SLEEP, SW_STNDBY, SNOOZE, DEEP_SW_STANDBY</td>
<td>SLEEP, SW_STNDBY, SNOOZE, DEEP_SW_STANDBY</td>
</tr>
</tbody>
</table>

Transitioning to the different LPMs is accomplished by using the RA LPM HAL driver API R_LPM_LowPowerModeEnter. More details of this API can be found in the FSP User's Manual.

1.5 Use Case: Operable Long Timer in Software Standby and Deep Software Standby Modes

The Operable Long timer in Software Standby and Deep Software Standby modes require a timer that can operate in a Low Power Mode. The count source of the timer is another element that should be considered carefully as well. In Renesas RA MCUs, the 16-bit Asynchronous General-Purpose Timer channel 0 (AGT0) and 16-bit channel 1 (AGT1) can be used in cascade mode to create an 32-bit timer. In the cascade mode, AGT0 underflow interrupt will trigger the counter of AGT1, AGT0 count source can be the sub-clock oscillator or LOCO clock, which are available in both Software Standby and Deep Software Standby modes. The AGT1 Underflow interrupt is used to wake the MCU up from LPM.

The maximum period of the 16-bit AGT timer channel 0 with the Sub-Clock count source running at 32.768 kHz is approximately 2.0 seconds. The Operable Long timer with two AGT timer channels in cascade mode will have a maximum period of approximately 2184.5 hours with a timer resolution of 30.517 µs.

Note: If a longer wakeup time is required, the RTC can be used via the RTC alarm, but here the resolution of the timer is limited to 1 second.

2. LPM HAL Module

The LPM HAL module in FSP provides a method to include the LPM driver into the application and to configure them for different modes. It also allows configuring different trigger/cancel signals as required for LPM activation/cancellation. FSP also provides essential APIs to configure and place the MCU in Low Power Modes. It supports the following Low Power Modes:
• Deep Software Standby mode (on supported MCUs)
• Software Standby mode
• Sleep mode
• Snooze mode

It also supports reducing power consumption when in deep software standby mode through internal power
supply control and by resetting the states of I/O ports.

3. FSP Configuration

When developing an FSP application in e² studio, first configure the FSP using the RA Configurator. To
properly configure the FSP, you must have detailed knowledge of both the software design that you will be
implementing, along with the specific hardware it will be running on. For the hardware, this includes the types
of peripherals to be used on the hardware, and the pins they are mapped to, internal or external to the MCU.
From the software perspective, you need to add the HAL modules for the peripherals you use and decide
how many threads will be used, and what additional software objects like semaphores, queues, and so on
that each thread will require. Once you have this information, you will be ready to successfully configure
the FSP for your specific application needs.

In an application using FSP, the FSP configuration is stored in a file named `configuration.xml`. Double
clicking on this file brings up the RA Configuration tab for the project.

![Figure 3. configuration.xml on the Project Plane](image)

When a project is built from scratch, this configuration tab is where you will perform the initial configuration of
the FSP. As you can see in Figure 4, the RA Configuration pane contains a Summary screen highlighting
the items you may configure, along with a scrolling window that lists all the software components currently
selected for this project. Below this scrolling window are tabs that allow you to tailor the FSP to the needs of
your specific application. More details on the use of the FSP configurator can be found in the FSP user’s
manual.

For the purposes of this application note, we will highlight a few of the details of the FSP properties such as
the r_lpm driver, r_rtc driver, and r_agt driver modules as they are key components operated in the use
cases provided in the application.

When you have configured the project appropriately, click the Generate Project Content, the green arrow
button above the summary screen, to build all the auto-generated files necessary to implement the
components you defined.
3.1 Components Tab

Even though the **Components** tab is the last tab showing, it is important to visit and verify the configured components are checked against the desired FSP version. Components are automatically selected when the modules are added in the **Stack** tab specific to the application. As the final step to verify the components selected, it is a good process to verify these selections are checked in the **Components** tab. One of the advantages of the FSP is that it will only compile the components you choose, thereby reducing the size of your overall application. As shown in Figure 5, components are broken down into seven categories.

![Figure 5. Components Tab Categories](image-url)

You may expand any of the categories by clicking the arrow to the left of the category name. The following table highlights the selections used for the LPM applications.
Table 4. Components Used in the LPM Applications

<table>
<thead>
<tr>
<th>Category</th>
<th>Component</th>
<th>Version</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BSP</td>
<td>ra6m3_ek</td>
<td>1.2.0</td>
<td>RA6M3-EK Board Support Package Files</td>
</tr>
<tr>
<td>CMSIS</td>
<td>CoreM</td>
<td>5.6.0</td>
<td>Arm CMSIS Version5 - Core (M)</td>
</tr>
<tr>
<td>Common</td>
<td>fsp_common</td>
<td>1.2.0</td>
<td>Board Support Package Common Files</td>
</tr>
<tr>
<td>HAL Drivers</td>
<td>r_cgc</td>
<td>1.2.0</td>
<td>Clock Generation Circuit</td>
</tr>
<tr>
<td></td>
<td>r_ioprt</td>
<td>1.2.0</td>
<td>I/O Port</td>
</tr>
<tr>
<td></td>
<td>r_lpm</td>
<td>1.2.0</td>
<td>Low Power Modes</td>
</tr>
<tr>
<td></td>
<td>r_icu</td>
<td>1.2.0</td>
<td>External Interrupt</td>
</tr>
<tr>
<td></td>
<td>r_gpt</td>
<td>1.2.0</td>
<td>General PWM Timer</td>
</tr>
<tr>
<td></td>
<td>r_agt</td>
<td>1.2.0</td>
<td>Asynchronous General-Purpose Timer</td>
</tr>
<tr>
<td></td>
<td>r_rtc</td>
<td>1.2.0</td>
<td>Real Time Clock</td>
</tr>
</tbody>
</table>

3.2 Stacks Tab
The Stacks tab is where you can add and configure the threads that the FSP automatically creates for your application. You define a new thread by clicking the button and then entering a unique name for your new thread. Once you add a new thread, you must define the modules that the thread will use along with any thread objects that will be used by your application thread.

As an example, if you click the Stacks tab and then single click on the HAL/Common thread, you should see something like the screen capture shown in Figure 6. This shows that the application requires multiple drivers, for example, the r_lpm driver which is the driver for Low Power Modes of Renesas RA MCU. The LPM applications do not use RTOS, hence there is only one HAL/Common thread is available in this type of application.

![Figure 6. Drivers Usage in LPM Application](image-url)

You can add additional modules to a thread by clicking the button. As an example, Figure 7 shows you how to add a AGT timer. The timer is added by choosing (+) New Stack > Driver > Timers > Timer Driver on r_agt.

If you pick a module that you have not preselected, the appropriate component for the module will be automatically selected by FSP for you. If the configurator tool detects errors due to incorrect settings with the module addition, it presents the module with an error. You may examine the errors by hovering over the module name.
3.3 Module Configuration

Once you have added a module to your project, you need to configure its properties. The properties are dependent on the module(s) that you have added. Use the Properties tab to configure them.

3.3.1 LPM Configuration

The LPM applications add the r_lpm driver module as the main component to configure the Renesas RA MCUs in Low Power Modes, for Sleep, Software Standby, Snooze, and Deep Software Standby. The main settings of Low Power Modes configures the trigger/cancel sources, and internal power supply conditions for Deep Software Standby mode.

3.3.1.1 Activation and Cancelation Sources

The cancelation source of an LPM will wake up the MCU from the specific LPM. The request of the Snooze mode puts the MCU into Snooze mode from Software Standby mode. These sources are interrupt sources. Refer to the Low Power Modes section in Renesas RA MCU Hardware Manual for more details on what interrupts are available in the LPM.

Table 5 shows the activation and cancelation sources using in the LPM applications.

**Table 5. Activation and Cancelation Source Configuration**

<table>
<thead>
<tr>
<th>Category</th>
<th>Interrupt Source</th>
<th>Application</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Request Source</td>
<td>AGT1_AGTI</td>
<td>Clock Changing and LPM Transition</td>
<td>AGT Channel 1 Underflow Interrupt</td>
</tr>
<tr>
<td>End Source</td>
<td>AGT1_AGTI</td>
<td>Clock Changing and LPM Transition</td>
<td>AGT Channel 1 Underflow Interrupt</td>
</tr>
<tr>
<td>Wake/Cancel Source</td>
<td>AGT1_AGTI</td>
<td>Operable Long Timer</td>
<td>AGT Channel 1 Underflow Interrupt</td>
</tr>
<tr>
<td></td>
<td>PORT_IRQ13-DS</td>
<td>Clock Changing and LPM Transition</td>
<td>External Interrupt 13-DS</td>
</tr>
<tr>
<td></td>
<td>RTC_ALM</td>
<td>Clock Changing and LPM Transition</td>
<td>RTC Alarm Interrupt</td>
</tr>
</tbody>
</table>

3.3.1.2 Sleep Mode Configuration

Since Sleep mode is canceled by any interrupts, there is no need to set up a cancel source for this mode in driver r_lpm configuration, as long as there is at least one source of interrupt active in the system. If the LPM application is using RTOS, the Systick timer must be stopped before entering the Sleep mode because the
Systick interrupt will wake up the MCU. When you are using the RTOS, if the Systick timer is stopped, it must be restarted after waking up for the proper RTOS kernel operation.

3.3.1.3 Software Standby Mode Configuration

Figure 8 shows how the LPM may be configured to exit Software Standby mode with AGT1 underflow interrupt as the wake source.

![Figure 8. Software Standby Properties Configuration using the Properties Tab](image)

3.3.1.4 Snooze Mode Configuration

The Snooze Request source triggers a transition from Software Standby mode to Snooze mode. The Snooze End source cancels Snooze Mode and transition the MCU back to Software Standby mode.

The wake source of Software Standby mode will wake up the MCU from both Software Standby and Snooze modes.

Figure 9 shows how to configure Snooze mode with AGT1 Underflow as both Snooze Request and End sources.
3.3.1.5 Deep Software Standby Configuration

Figure 10 shows Deep Software Standby Mode with AGT1 underflow interrupt as cancel source.

If IRQ is used to cancel the Deep Software Standby Mode, you need to use the IRQn-DS interrupts.

In Deep Software Standby mode, the “Internal power supply control in deep software standby mode” provides an option to maintain the internal power supply to the peripherals or cut down the internal power supply to most of the internal peripherals. For example, Maintain the internal power supply is needed to keep AGT timer running in the Operable Long Timer application as shown in Figure 11.
This is the option used in this application example. For other available options, refer to the FSP User manual and MCU Hardware user manual.

### 3.3.2 Timer Configuration

#### 3.3.2.1 RTC Configuration

The Real Time Clock (RTC) is one of the peripherals which can operate in all the Low Power Modes. The RTC is primarily used for time keeping, which updates the time independent of MCU in LPM. In this application, the RTC is used for keeping track of the time in the Clock Changing and LPM Transition application and to wake up from Software Standby mode via the RTC alarm interrupt. RTC uses the LOCO as the clock source in the application.

The application displays the RTC time when the MCU transition from LPM to Normal mode. This gives the indication of the how long the MCU was in LPM and latest time info.

The following Figure 12 shows the configurations of RTC for Time and Alarm. Alarm is used as trigger in the LPM. The clock source of RTC is LOCO, which is available in Low Power Modes.

#### 3.3.2.2 AGT Timer Configuration

As mentioned earlier, the Operable Long Timer application in LPM uses AGT timer channel 0 (AGT0) and AGT timer channel 1 (AGT1) in cascade mode to create a 10-second operable long timer. In this mode, AGT0 underflow interrupt will trigger the counter of AGT1.

The following figures show the configurations of AGT0 and AGT1. The count source of AGT0 is the Sub-Clock, which is available in Low Power modes, and the count source for AGT1 is the AGT0 underflow interrupt.

AGT0 is configured in a periodic mode that generates underflow interrupt every second shown in Figure 13.
Figure 13. AGT0 Properties Configuration using the Properties Tab

AGT1 is also configured in a periodic mode with raw count as the period unit shown in Figure 14.

Figure 14. AGT1 Properties Configuration using the Properties Tab
3.4 Pin Configuration

The FSP application can support multiple pin configurations. In this application we use two different pin configurations, one for active mode of operation and other for power saving mode operation. Refer to the Renesas Flexible Software Package (FSP) User’s Manual on how to configure the FSP Pin Configuration.

3.4.1 Pin Configuration in Normal Mode

The pin configuration in normal mode is the MCU pin functionalities that you want to use in full operating condition. Figure 15 shows the pin configuration of the EK-RA6M3 kit which is used in normal mode.

![Figure 15. Pin Configuration of EK-RA6M3 Kit in Normal Mode](image)

3.4.2 Pin Configuration in LPM

Use the pin configuration in Low Power Modes to reduce power consumption by disabling unused pins, which puts them in the input mode. Refer to the “Handling of Unused Pins” section of the MCU Hardware user manual for more details.

Figure 16 shows the pin configuration of the EK-RA6M3 kit in LPM named RA6M3-EK-LPM.pincfg with unused pins disabled. You may observe that most pins are disabled, except for the IRQ pins, and CGC pins, which are used to wake up the MCU and provide clock input through the XTAL and EXTAL pins.
4. Application Architectures

The following figure, Figure 17, shows transitions between LPM and Normal mode in LPM applications. The WFI instructions activate LPM, the configured interrupt such as AGT1 underflow interrupt or external interrupt IRQ13-DS cancels LPM and wakes up the MCU.

**Note:** The transition from the Normal Mode to the Deep Software Standby mode is handled internally by the LPM driver via the Software Standby mode with DPSBYCR.DPSBY bit set to 1.
4.1 Clock Changing and LPM Transition

This application demonstrates the use case where the clock source to the CPU is changed at runtime. It also demonstrates entering and exiting the different LPM using the API provided by the FSP. Without the availability of the APIs, a developer would need to configure the registers related to LPM and CGC manually, thereby adding to development timeline.

The following Figure 18 shows the different clock sources and the associated LPM used in the application and different transition states. The application implemented using the event driven mechanism. In this event driven system, events can be user driven event or system generated event, which are used as the input to the finite state machine. Two separate state transition tables are used here in the application.

- Clock transition table \(\text{clock\_transition\_table}\)
- LPM Transition table \(\text{lpm\_transition\_table}\).

The transition table has the list of actions to be performed based on the events received. For instance, when user event “Button Press - Long” is received upon power on reset, the finite state machine will start running and change the clock to HOCO. If user event “Button Press – Short” is received, the finite state machine will switch to the LPM state machine and start the LPM operation.
In the above Figure 18, the blue colored oval shaped blocks (B) though (F) represent the different clock states used in the application, and (b) through (f) labeled arrows are different transition path it takes when changing the clock.

The yellow blocks are the different LPM states as applicable to the MCU, and (1) through (8) numbered arrows, represents the transition path at the different Low Power Modes.

**Note:** The dotted block represents the Clock change Mode transition on the left and LPM Transition on the right as shown in Figure 18

**Note:** The Deep Software Standby Mode and Snooze Mode is entered via Software Standby mode. For the Snooze and Deep Software Standby mode, from the application perspective the MCU and LPM drivers handles the Software Standby mode internally, while it is configured for the Snooze or Deep Software Standby Mode.

**Note:** The clock can only be changed in the Normal Mode. Changing clock in the LPM mode is not allowed.
Note: In the above Figure 18, for low end RA devices (RA2A1) Sleep, Software Standby, and Snooze modes are supported. For the higher end RA devices (such as RA6M3) in addition to the LPM supported by low end devices, it also supports the Deep software standby mode.

Events used in this application are as listed in the Table 6.

Table 6. Events used for the Clock and LPM transitions

<table>
<thead>
<tr>
<th>List of Event</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EV_PB_SHORT_PR</td>
<td>User push button event – “Short Press” – held for 1-2 seconds</td>
</tr>
<tr>
<td>EV_PB_LONG_PR</td>
<td>User push button event – “Long Press”– held for 4-6 seconds</td>
</tr>
<tr>
<td>EV_PERIODIC_TIMER</td>
<td>AGT1 timer event generated by timer overflow</td>
</tr>
<tr>
<td>EV_RTC_ALARM</td>
<td>RTC Alarm Interrupt generated based on the configured time.</td>
</tr>
<tr>
<td>EV_POWER_ON_RESET</td>
<td>Power on Reset event,</td>
</tr>
<tr>
<td>EV_POWER_ON_RESET_DSSBY</td>
<td>Reset from Deep Software Standby Mode.</td>
</tr>
</tbody>
</table>

4.2 RTC Timer Operation in LPM

An additional feature is part of the clock changing and LPM transition application, which showcases the running RTC peripheral during the LPM. Even when the CPU and most components in the MCU enter Low Power Modes and cease operation, the RTC clock and its timer operates independently. Updated RTC time information are displayed on to the RTT when it transitions back to the Normal mode.

RTC Periodic/Alarm Interrupts can be used as signals for transitioning to different LPMs. In this LPM application project, RTC Alarm interrupt is used for canceling the Snooze mode/Software Standby mode and revert to the Normal mode.

Note: The RTC Periodic interrupt can also be used for this event. The RTC Periodic interrupt has a maximum period of 2 seconds. For demo purposes, we are avoiding this to showcase the step by step transition and not to be limited to the 2 seconds. However, the RTC alarm can be configured to the desired seconds.

4.3 Operable Long Timer in Software Standby and Deep Software Standby Modes

The Operable Long Timer Application uses LPM configurations in Software Standby and Deep Software Standby modes. It disables unused clock and IO ports before entering LPM with the WFI instruction, then places IO ports back to normal operating condition after waking up as shown in Figure 19.

LPM is canceled by the Operable Long Timer underflow interrupt, which is created by using AGT0 and AGT1 in cascade mode.
5. Application Code Highlights

5.1 Clock Source Setup

5.1.1 Handle On-Chip Modules in LPM to Reduce Power Consumption

Oscillators and on-chip modules may be started automatically after MCU reset and be still running in LPM modes. Therefore, to reduce MCU power consumption, you should disable these modules before entering LPM.

Unused IO ports in LPM other than in Deep Software Standby mode should be put into input mode before entering the LPM by using the `g_bsp_pin_lpm_cfg`, which is generated from the `RA6M3-EK-LPM.pincfg`. The following code fragments highlight the steps in the application to reduce MCU power consumption before placing MCU in LPM and restoring the IO port settings after waking it up.
Open and configure the LPM.

```c
/* Open LPM instance*/
err = R_LPM_Open(&lpm_ctrl_instance_ctrls[g_lpm_transition_pos], &lpm_ctrl_instance_cfgs[g_lpm_transition_pos]);
/* Handle error */
if (FSP_SUCCESS != err)
{
    APP_ERR_TRAP(err);
}
```

**Figure 21. Open and Configure the LPM**

Place the MCU in the LPM.

```c
/* Enter LPM mode */
err = lpm_mode_enter(g_lpm_transition_sequence[g_lpm_transition_pos]);
/* Handle error */
if (FSP_SUCCESS != err)
{
    /* Turn on user LED to indicate error occurred*/
    R_IIMPORT_PinWrite(&iimport_ctrl, leds.p_leds[LED_NO_0], BSP_TO_LEVEL_HIGH);
    APP_ERR_TRAP(err);
}
```

**Figure 22. Place the MCU in the LPM**

Put IO port in normal mode by configuring them using the `g_bsp_pin_cfg` generated from the `RA6M3-EK.pincfg` after exiting from LPM.

```c
/* Put IO port configuration back to user's selections */
err = R_IIMPORT_PinsCfg(&iimport_ctrl, &g_bsp_pin_cfg);
/* Handle error */
if (FSP_SUCCESS != err)
{
    APP_ERR_TRAP(err);
}
```

**Figure 23. Place IO Ports in Normal Mode**

In this application example for the Deep Software Standby mode, the IO port must be released from IOKEEP state after MCU reset.
5.1.2 Change System Clock at Run-Time

When the MCU power up, the default clock will be the configured clock as part of the BSP. Selecting the clock for the application is done through the FSP configurator using the clock tree, which is available in the clocks tab.

In the changing clock use case application, the current running system clock is read via the R_CGC_SystemClockGet API and a new system clock source is configured. The following code is used for reading the currently running system clock shown in Figure 26.

```c
err = R_CGC_SystemClockGet(&g_cgc0_ctrl, &sys_clock_source, &sys_divider_cf);
if (FSP_SUCCESS != err)
{
    APP_ERR_TRAP(err);
}
```

In order to configure the new source for the system clock, proper divisors are required so that the peripherals get the permitted range of frequencies. Users need to calculate this based on the peripherals used in the application. The new system clock is configured via the API R_CGC_ClocksCfg.

Also, the clock sources need to be started if they are not already running. This is done using the API R_CGC_ClockStart as shown in the Figure 27.

```c
if (CGC_CLOCK_SUBCLOCK == sys_clock_source)
{
    new_clk.source_clk = CGC_CLOCK_SUBCLOCK;
    err = R_CGC_ClockStart(&g_cgc0_ctrl, CGC_CLOCK_SUBCLOCK, &new_clk);
    if (FSP_SUCCESS != err)
    {
        APP_ERR_TRAP(err);
    }
}
```
6. Importing and Building the Project

To bring the applications into the e² studio ISDE, follow these steps:

1. Launch e² studio ISDE.
2. In the workspace launcher, browse to the workspace location of your choice.
3. Close the Welcome window.
4. In the ISDE go to File > Import.
5. In the Import Dialog Box pick Existing Projects into Workspace.
6. Select the Root directory of your workspace (where you placed the project).
7. Select the project you wish to import and click Finish.
8. Click on Generate Project Content on the FSP configurator window.
9. Now build the project.

7. Running Applications

To connect and run the code, follow these steps:

7.1 Board Setups

The EK-RA6M3 kit contains a few switch settings which must be configured prior to running the application associated with this application note. In addition to these switch settings, it has a USB debug port and connectors to access the J-Link® programming interface.

Table 7. Switch settings for EK-RA6M3

<table>
<thead>
<tr>
<th>Switch</th>
<th>Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>J8</td>
<td>Jumper on pins 1-2</td>
</tr>
<tr>
<td>J9</td>
<td>Open</td>
</tr>
</tbody>
</table>

7.2 Downloading the Executables

The executable file may be programmed into the target MCU through any one of three means.

7.2.1 Using a debugging interface with e² studio

Instructions to program the executable binary are found in the latest RA FSP User Manual. See Section Starting Development > e² studio ISDE User Guide > Tutorial: Your First RA MCU Project > Debug the Blinky Project.
This is the preferred method for programming as it allows for additional debugging functionality available through the on-chip debugger.

### 7.2.2 Using J-Link tools

SEGGER J-Link Tools such as J-Flash, J-Flash Lite, and J-Link Commander can be used to program the executable binary into the target MCU. Refer to User Manuals UM08001 and UM08003 on [www.segger.com](http://www.segger.com).

### 7.2.3 Using Renesas Flash Programmer

The Renesas Flash Programmer provides usable and functional support for programming the on-chip flash memory of Renesas microcontrollers in each phase of development and mass production. The software supports all RA MCUs and software user’s manual is available online (Renesas Electronics Corporation, n.d.).

### 7.3 User Interface

The user interface to interact with the application is shown below. The Operable Long Timer application uses the LED and user push button switch. Whereas the Clock Changing and LPM Transition application uses the RTT interface in addition to the LED and push button switch.

#### 7.3.1 LED Indication

**7.3.1.1 Clock Changing and LPM Transition**

The Clock Changing and LPM Transition application uses the LED1 to indicate the board initialization status, error condition, and normal mode operation. In the Sleep, Software Standby, Snooze, and Deep Software Standby modes, LED1 will be turned off. In the Normal mode, this LED1 will blink every second. If any error condition occurs the LED1 will be turned ON.

**7.3.1.2 Operable Long Timer**

Operable Long Timer application uses the LED1 to indicate the normal mode. In the Normal mode, this LED1 will blink. In the Software Standby and Deep Software Standby modes, LED1 will be turned off.

#### 7.3.2 User Push Button Input

**7.3.2.1 Clock Changing and LPM Transition**

Push button switch S1 input is mainly used for transition to different MCU clocks and transition to different LPM. For the Clock Changing and LPM Transition application, the same switch has dual functionality. If the switch is held under 1-2 seconds, it is considered a short press. If the switch is held for 3–6 seconds, it is considered a long press. A long press event is used for changing the Clock source dynamically and a short press event is used for the LPM mode transition.

**Note:** A long press event during the LPM will not change the system clock source to a different clock, but instead has a different role: to exit the LPM and go back to the Normal mode.

**7.3.2.2 Operable Long Timer**

In the application, pressing the push button switch S1 will set the Software Standby and Deep Software Standby modes. The AGT1 underflow interrupt will cancel the LPM in 10 seconds.

#### 7.3.3 RTT Console

The RTT console comes in handy to view the application messages while running/debugging the application. While using the RTT console the debugger script for the LPM needs to be selected as shown in Figure 29.
7.4 Debugging Low Power Modes

By default, it is not possible to debug the low power modes of RA device. If an application tries to enter sleep mode, pending a peripheral interrupt to wake it, then this will not happen as it will be woken almost immediately by a debug interrupt.

If the application tries to enter software or deep software standby modes, then connection between the CPU and the IDE will be lost, closing the debug session within the IDE.

However, if the supplied debug script is specified then it will be possible to debug the low power modes. For the purpose of demonstrating the LPM, these scripts are used. Note, that even though this will allow you to develop your application, it will not allow you to measure accurate Icc figures, as you will be measuring the Icc of the on-chip debug circuit. Once you have created your low power application, accurate Icc figures can be measured with the OCD disabled.

The low power script also allows for the RTT application to be used. While debugging the application, configure the debugger as shown in Figure 30. With these modifications one can use the RTT without getting disconnected during the LPM.

Note: The script is attached as part of this project and the debugger is pointing to the same location.
7.5 Steps to Run the Application

The following table shows the steps to run the Clock Changing and LPM Transition application. On power up, when the board is connected to RTT, it will display the welcome message. To change the clock/LPM and run through the different use cases in this application, use the following tables as reference. These tables have the list of events, current mode, and refers the new transition states and the expected outcome.

### 7.5.1 Clock Changing:

The user button has two functions, based on how long you press and hold it. In the clock changing mode, the user button pressed and held for 3-6 seconds selects the different system clocks for the MCU. Whereas the user button pressed and held for 1-2 seconds exits the clock changing mode and enters into LPM for the configured clock. The details are as shown in Table 8.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MOSC (Main Oscillator)</td>
<td>Changes the System Clock to HOCO</td>
<td>Exits the Clock State Machine and enters the LPM State Machine for the configured clock MOSC.</td>
</tr>
<tr>
<td></td>
<td>HOCO (High speed on-chip oscillator)</td>
<td>Changes the System Clock to MOCO</td>
<td>Exits the Clock State Machine and enters the LPM State Machine for the configured clock HOCO.</td>
</tr>
<tr>
<td></td>
<td>MOCO (Medium speed on-chip oscillator)</td>
<td>Changes the System Clock to LOCO</td>
<td>Exits the Clock State Machine and enters the LPM State Machine for the configured clock MOCO.</td>
</tr>
<tr>
<td></td>
<td>LOCO (Low speed on-chip oscillator)</td>
<td>Changes the System Clock to SOSC</td>
<td>Exits the Clock State Machine and enters the LPM State Machine for the configured clock LOCO.</td>
</tr>
<tr>
<td></td>
<td>SOSC (Sub Oscillator)</td>
<td>Changes the System Clock to MOSC</td>
<td>Exits the Clock State Machine and enters the LPM State Machine for the configured clock SOSC.</td>
</tr>
</tbody>
</table>

### 7.5.2 LPM Transition

In the LPM application, the user switch has two functionalities based on how long you pressed, held, and released it. If the MCU is in the LPM modes (Sleep or Deep Software Standby) and the user button is
pressed, held, and released, MCU exits the LPM mode and enters the Normal mode whether it was a short press or long press. Whereas the behavior is different when the MCU is in Normal mode.

When the MCU is in Normal mode, if the user button is pressed, held, and released for 1-2 seconds, MCU exits the Normal mode and enters (Sleep or Software Standby or Snooze or Deep software standby) modes depending on the previous transition states. Whereas if the switch is pressed, held, and released for 3-6 seconds, MCU exits the LPM transition setting mode and enters the Clock setting mode. Also the user switch has no effect during the LPM modes (Software Standby or Snooze).

Timer events cancel the Software Standby and Snooze modes and puts the MCU back to Normal mode.

The details of the switch events and timer events are tabulated in Table 9.

Table 9. LPM Transition Table

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal</td>
<td>Exits the LPM transition State Machine and enters the Clock Mode State Machine</td>
<td>Enters the Sleep mode</td>
<td>Not Applicable</td>
</tr>
<tr>
<td>Sleep</td>
<td>Exits the Sleep mode and enters the Normal mode</td>
<td>Exits the Sleep mode and enters the Normal mode</td>
<td>Not Applicable</td>
</tr>
<tr>
<td>Normal (From Sleep)</td>
<td>Exits the LPM transition State Machine and enters the Clock Mode State Machine</td>
<td>Exits the Normal mode and enters the Software Standby mode</td>
<td>Not Applicable</td>
</tr>
<tr>
<td>Software Standby</td>
<td>Not Applicable</td>
<td>Not Applicable</td>
<td>Exits the Software Standby mode and Enters the Normal mode</td>
</tr>
<tr>
<td>Normal (From Software Standby)</td>
<td>Exits the LPM transition State Machine and enters the Clock Mode State Machine</td>
<td>Exits the Normal mode and enters the Snooze with Software Standby mode</td>
<td>Not Applicable</td>
</tr>
<tr>
<td>Snooze with Software Standby</td>
<td>Not Applicable</td>
<td>Not Applicable</td>
<td>Exits the Snooze with Software Standby mode and Enters the Normal mode</td>
</tr>
<tr>
<td>Normal (From Snooze)</td>
<td>Exits the LPM transition State Machine and enters the Clock Mode State Machine</td>
<td>Exits the Normal mode and enters the Deep Software Standby mode</td>
<td>Not Applicable</td>
</tr>
<tr>
<td>Deep Software Standby</td>
<td>Exits the Deep Software Standby mode and resets the MCU</td>
<td>Exits the Deep Software Standby mode and resets the MCU</td>
<td>Not Applicable</td>
</tr>
</tbody>
</table>

7.5.3 Operable Long Timer

In the Operable Long Timer application, the user button is used to enter Software Standby and Deep Software Standby modes from Normal mode. AGT1 underflow interrupt is used to exit the LPM modes.

Table 10 shows the transition sequence and associated events used in the Operable Long Timer application.
### Table 10. LPM Transition Table in Operable Long Timer Application

<table>
<thead>
<tr>
<th>Low Power Modes</th>
<th>User Button – Press</th>
<th>Timer Event – AGT1_AGT1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal</td>
<td>Enters Software Standby Mode</td>
<td>Not Applicable</td>
</tr>
<tr>
<td>Software Standby</td>
<td>Not Applicable</td>
<td>Exits the Software Standby mode and enters the Normal Mode</td>
</tr>
<tr>
<td>Normal (From Software Standby)</td>
<td>Enters Deep Software Standby Mode</td>
<td>Not Applicable</td>
</tr>
<tr>
<td>Deep Software Standby</td>
<td>Not Applicable</td>
<td>Exits the Deep Software Standby Mode and resets the MCU</td>
</tr>
</tbody>
</table>

#### 7.6 Measure MCU Current

The following steps are required to measure MCU current on EK-RA6M3, which is supported by the LPM applications:

- Remove the R2 (resistor).
- Measure the voltage drops across R3 and calculate MCU's current (Icc); replace R3 by a bigger resistor if needed.

To measure the MCU's current (Icc) directly, connect a multimeter between the +3V3 and +3V3_MCU pins on the kit connectors after removing the R3 resistor.

#### 8. Migrating LPM Applications to Different MCU/Kit

Even though the LPM applications are created for the Renesas EK-RA6M3 kit, they are designed to easily migrate to other Renesas RA Kits. Refer to Table 1 of this document for more details about the supported Renesas RA kits. The following steps are basic procedure to port the project to supported Renesas RA kits.

Rename and import the projects to e² studio, for example, changing `OLT_Timer_App_EK_RA6M3` to `OLT_Timer_App_EK_RA6M2` as shown in Figure 31.
Figure 31. Rename and Import Project into e² studio
Open the project configuration and change the board BSP from EK-RA6M3 to EK-RA6M2 in the BSP tab.

Figure 32. Change Board BSP

Click at the Pins tab, select RA6M2-EK.pincfg and generate project content.

Figure 33. Select Default Pin Configuration

Create a new pin configuration file named RA6M2-EK-LPM.pincfg in the same project folder by copying and renaming the RA6M2-EK.pincfg.

Figure 34. Create a New Pin Configuration
Select the RA6M3-EK-LPM.pincfg and uncheck the Generate Data box to deselect the old pin configuration.

![Figure 35. Deselect the Old LPM Pin Configuration](image)

Select the newly created RA6M2-EK-LPM.pincfg and check the Generate Data box. You can disable unused pins and peripherals and generate the pin configuration for LPM, for example, disabling SCI7 as shown in Figure 36.

![Figure 36. Select the New Pin Configuration and Disable Unused Peripherals](image)
Figure 37 shows an example of placing P106 into input mode.

**Figure 37. Place Unused in Input Mode**

Change the r_icu driver configuration and other peripheral configurations if needed. This is necessary since the user pushbutton S1 used in the applications may be different from board to board.
Figure 38. Change Peripheral Configuration

When migrating the projects to a new board that is not listed in Table 1, you must modify the code to add board definition for the new board type. Similarly, add the new board type in other parts of the application as required. The sample code fragment of the board type being used in the application is as shown below in Figure 39.

```c
/* File Name : lpm_app_transition_sequence_config.h */
/* DISCLAIMER */

#ifndef LPM_APP_TRANSITION_SEQUENCE_CONFIG_H
#define LPM_APP_TRANSITION_SEQUENCE_CONFIG_H

/*-------------------- LPM_APP_TRANSITION_SEQUENCE_CONFIG_H ------------------------*/

/*@brief Pre-defined LPM Transition Sequence
There are 2 kinds of sequence, one supports RA96x device and the others support other RA devices
 ************************************************************/  
@if defined (BOARD_RA961_EK) || defined (BOARD_RA301_EK) || defined (BOARD_RA492_EK)  
app_lpm_states_t g_lpm_transition_sequence[] = {
  APP_LPM_SH_SUSPEND_STATE,    //< Sh Standby mode
  APP_LPM_DEEP_SH_STANDBY_STATE,    //< Deep Sh Standby mode
  APP_LPM_NORMAL_STATE    //< Normal mode
};
@endif defined (BOARD_RA961_EK) || defined (BOARD_RA301_EK) || defined (BOARD_RA492_EK)
app_lpm_states_t g_lpm_transition_sequence[] = {
  APP_LPM_SH_SUSPEND_STATE,    //< Sh Standby mode
  APP_LPM_NORMAL_STATE    //< Normal mode
};
#endif /* LPM_APP_TRANSITION_SEQUENCE_CONFIG_H */
```

Figure 39. Board Types Used in the LPM Application

Build, run and clean up the migrated project.
9. References

- Renesas RA MCU Datasheets: See http://renesas.com/ra and select the relevant MCU
- LPM Example Projects on Renesas RA GitHub: https://github.com/renesas/ra-fsp-examples
Website and Support

Visit the following vanity URLs to learn about key elements of the RA family, download components and related documentation, and get support.

RA Product Information  www.renesas.com/ra
RA Product Support Forum  www.renesas.com/ra/forum
RA Flexible Software Package  www.renesas.com/FSP
Renesas Support  www.renesas.com/support
## Revision History

<table>
<thead>
<tr>
<th>Rev.</th>
<th>Date</th>
<th>Description</th>
<th>Page</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.00</td>
<td>Sep.29</td>
<td></td>
<td>-</td>
<td>Initial version</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>


Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.

2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.

3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.

4. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.

5. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The intended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below.

   “Standard”: Computers, office equipment, communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.

   “High Quality”: Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; underwater repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user’s manual or other Renesas Electronics document.

6. When using Renesas Electronics products, refer to the latest product information (data sheets, user’s manuals, application notes, “General Notes for Handling and Using Semiconductor Devices” in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.

7. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.

8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.

9. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.

10. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.

11. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.

12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.

(Note2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

(Rev.4.0-1 November 2017)

Corporate Headquarters
TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Contact information
For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit:
www.renesas.com/contact/

Trademarks
Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.

© 2019 Renesas Electronics Corporation. All rights reserved.